Aalto University
School of Science

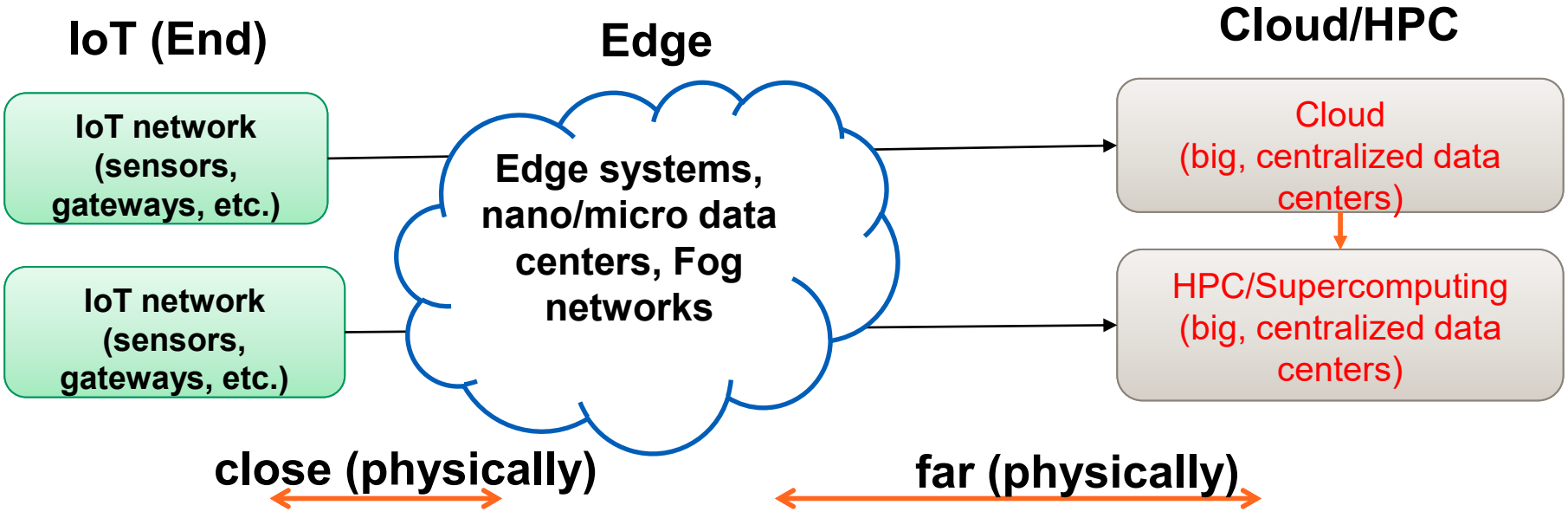
Machine Learning with Edge-centric Systems

Hong-Linh Truong
Department of Computer Science
linh.truong@aalto.fi, https://rdsea.github.io

Learning objectives

- Understand and analyze the relationship between edge computing and ML
- Explore and study basic concepts and issues when engineering ML in edge-centric systems
- Identify and work on ML optimization problems across levels of abstraction in edge-centric systems

IoT-Edge-Cloud



"Edge" is just an abstraction view

Edge computing

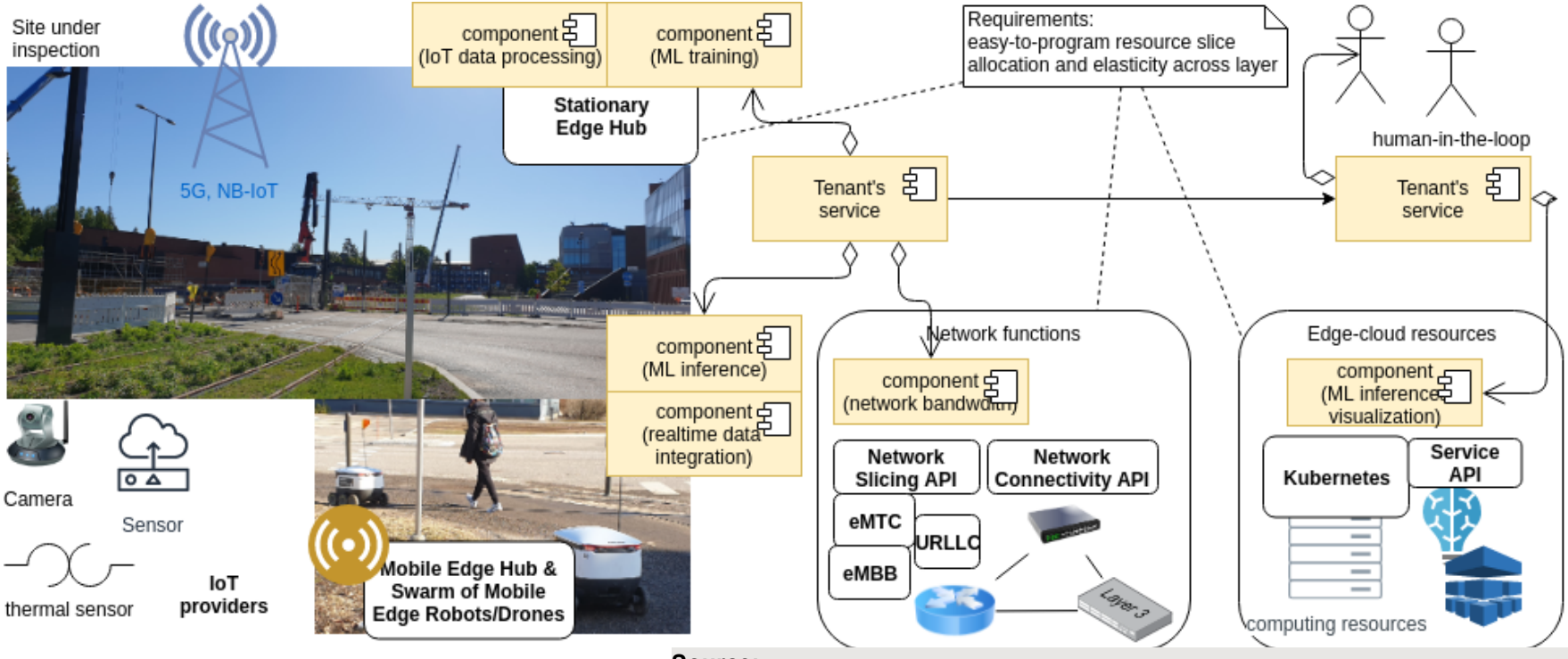
- **Distributed computing at the edge and end devices**
  - *many distributed low-end as well as a limited number of high-end devices/machines for different purposes*
- **Leveraging common technologies like in the cloud and specific ones**
  - *e.g., virtualization, container orchestration, messaging systems, storage/database, Web services*
- **But with different constraints**
- **Edge-centric systems**
  - edge systems but combined with the cloud and others

Edge computing

- **Computation/analytics can be done at the edge**
  - where data is generated/collected, close to the data sources
    - *next to IoT devices and sensing equipment*
  - many distributed (moving) locations, e.g., in the shopping center, in the car
- **Near real-time data processing and analytics needed in most situations**
- **High heterogeneity w.r.t system models, hardware architectures, network connectivities, and protocols**

Example of a scenario

Training/ML inferences: move from the cloud to the edge



In city blocks, villages, etc.

Source: https://www.researchgate.net/publication/361261810_Robustness_via_Elasticity_Accelerators_for_the_IoT-Edge-Cloud_Continuum

Why do we have to support ML/data analytics at the edge?

Machine learning/big data analytics in the edge

- **Many applications can benefit from ML/data analytics capabilities**
  - inferencing/classification in mobile devices/smart homes, healthcare
  - (near) real-time ML-based steering (autonomous cars, speech control, traffic controls)
  - (near) real-time anomaly detection and forecasting: fraud detection, fault detection, and incidents (e.g., in manufacturing, network operations, security monitoring)

Machine learning/big data analytics in the edge

- **Close to data sources → "data locality" benefits**
  - security & privacy
  - performance
  - customization/personalization
  - cost saving
  - working under no connectivity
- **But with many challenges!**

Aalto University
School of Science

Basic concepts/issues when engineering ML in edge systems

Very new area! a lot of ongoing research and development activities!

Things affecting robustness, reliability, resilience and elasticity

- **Network problems**
  - high latency, low-bandwidth, unreliable connectivity
- **Computation capabilities**
  - constrained processing power, a lot of specific chips and accelerators, and limited memory
- **Storage is not enough for big data**
- **V* issues in data**
  - out of distribution data, unlabeled data, time series data, streaming data
- **Energy/power usage of devices/machines**

Things affecting ML capabilities

- **Edge with hardware heterogeneity**
  - common hardware (e.g., AMD, Intel, ARM), SoC and microcomputers, microcontrollers
  - with/without common and AI-based accelerators like FPGA, GPU, and TPU

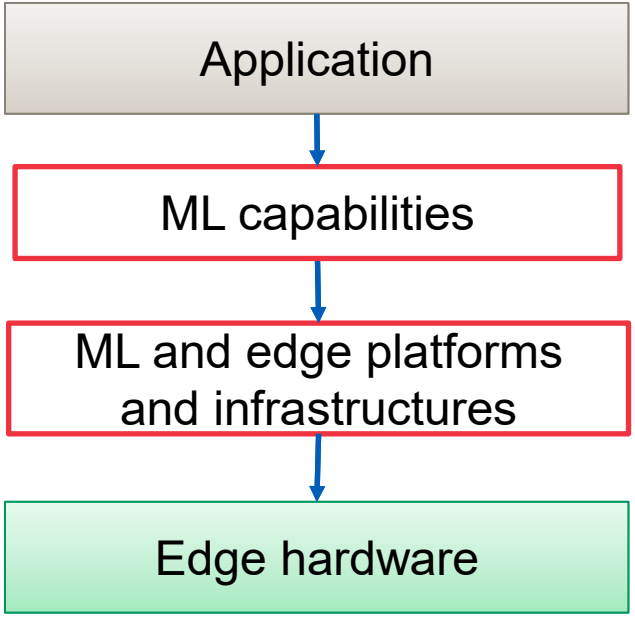
→ Requirements for certain types of ML might not be fulfilled: computation-intensive ML (e.g., video analytics)

Pervasive embedded edge devices

- Raspberry PI4
- Google Coral
- Jetson Nano
- Xilinx
- A huge number of MCUs (Microcontroller Units)

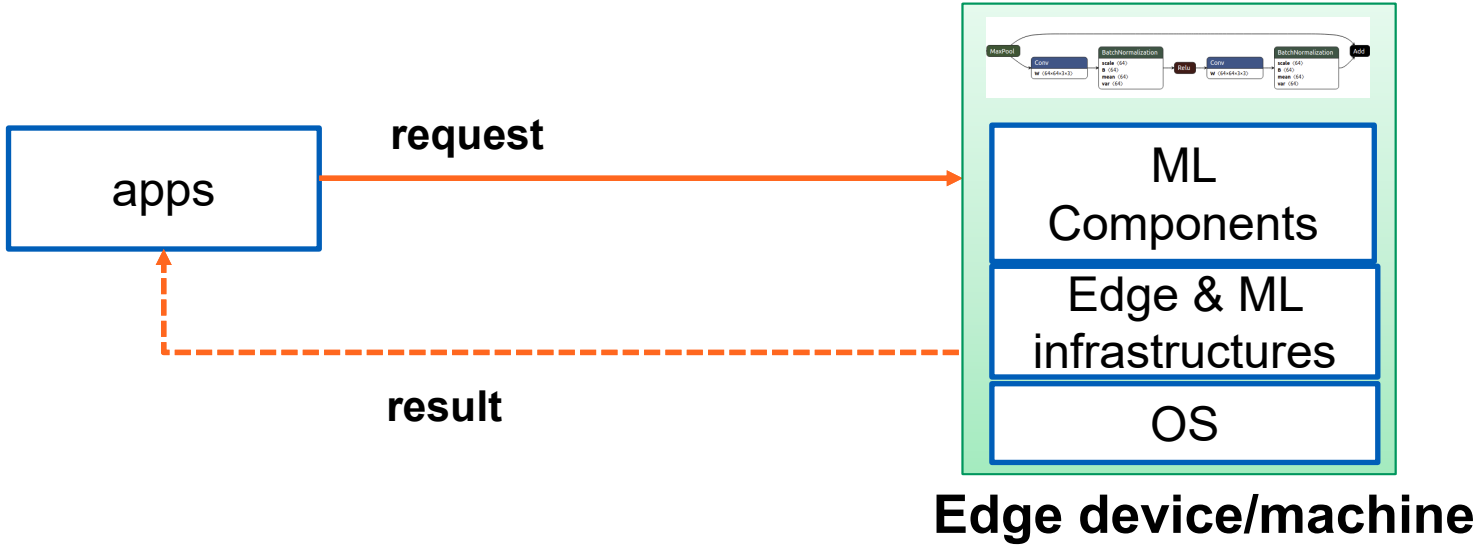
Interaction models in edge-cloud ML systems

Which components do what, and where are they?



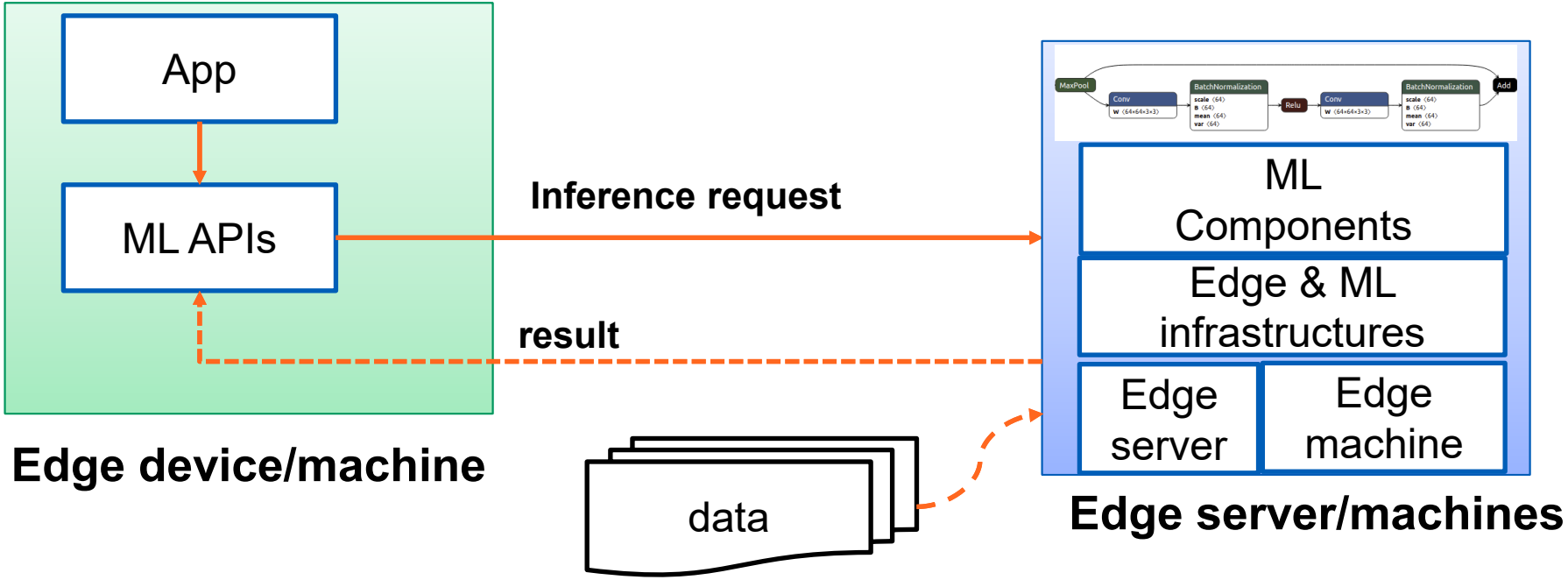
Interaction models

Standalone/in-device ML capabilities within independent devices



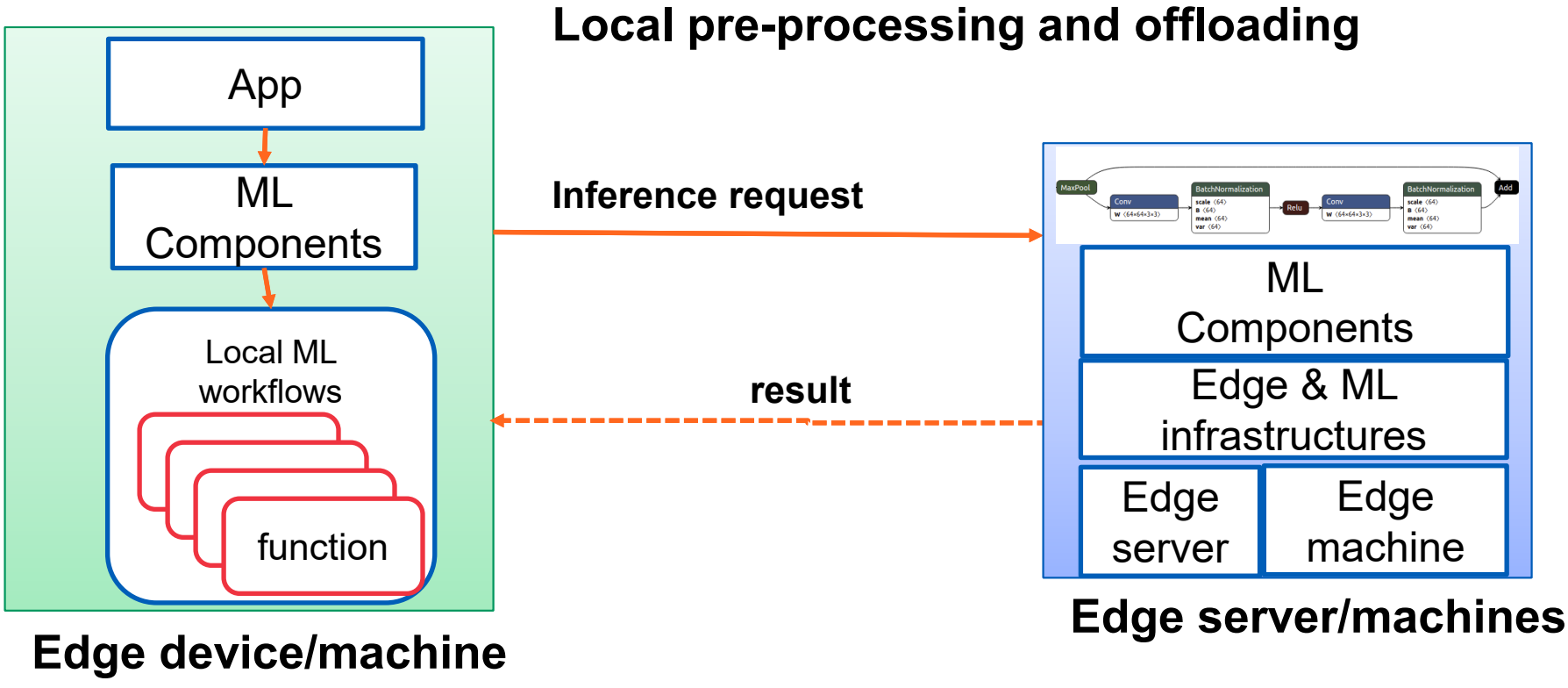
Interaction models

Common client-server model without local processing



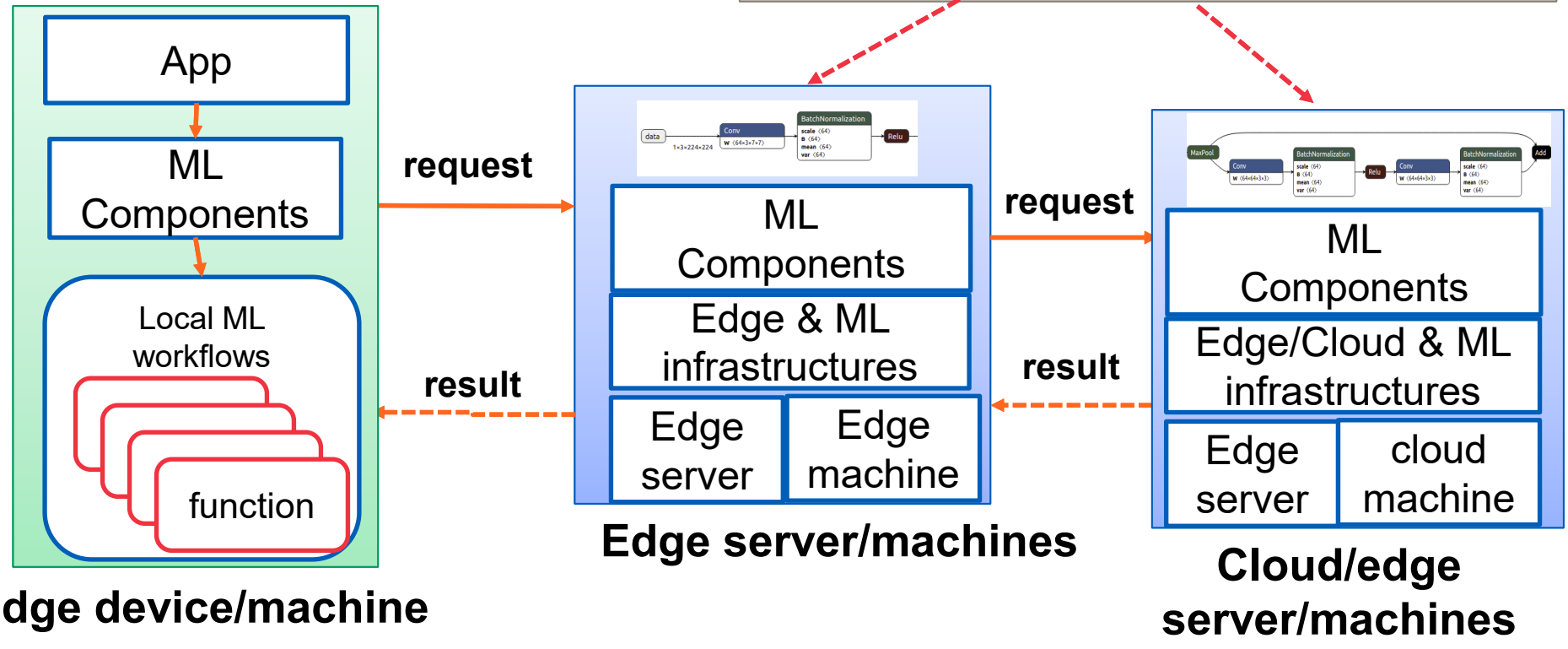
Can we use this for distributed training? Inferences?

Interaction models

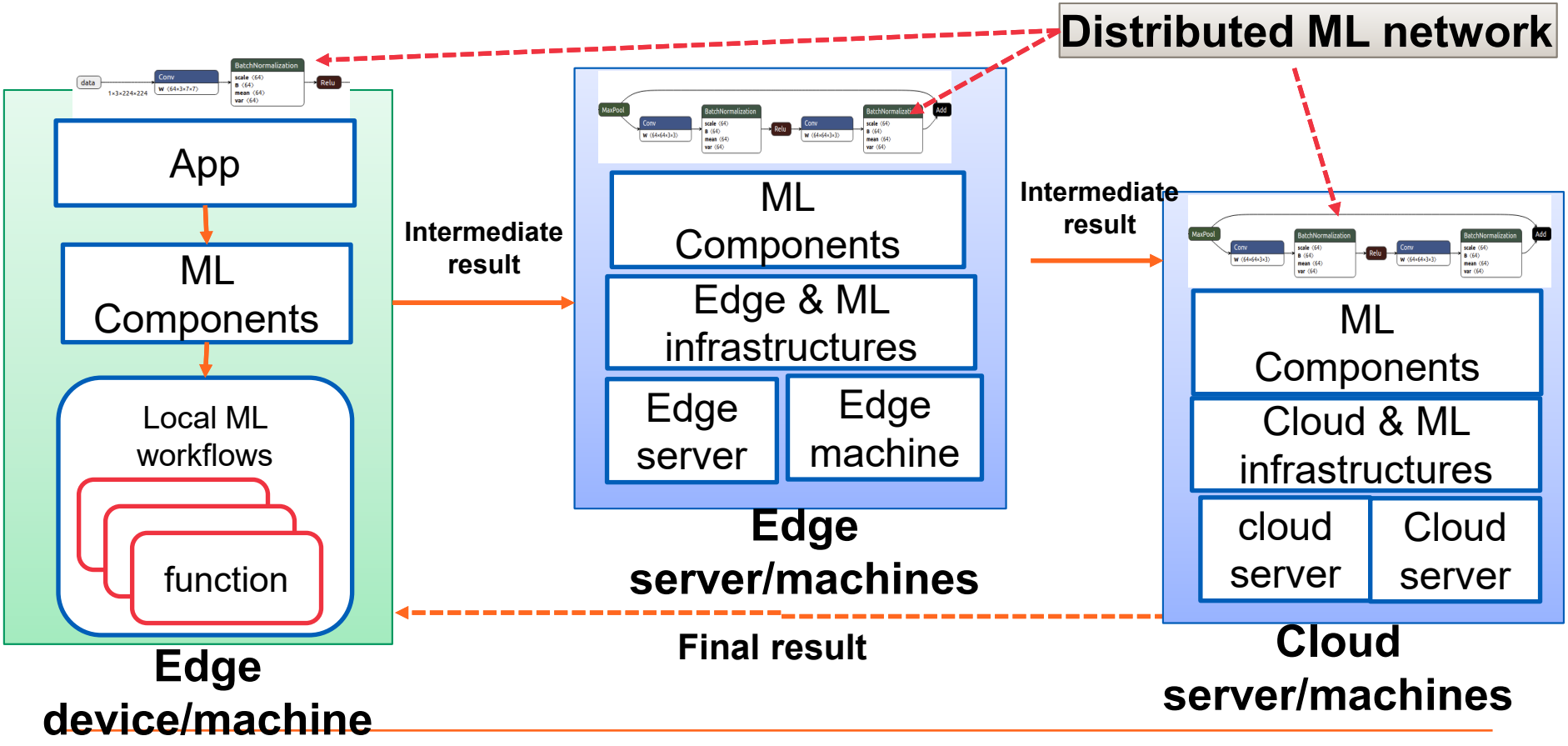


Interaction models

ML service chain/composition: distributed ML model instances/training in edge-cloud

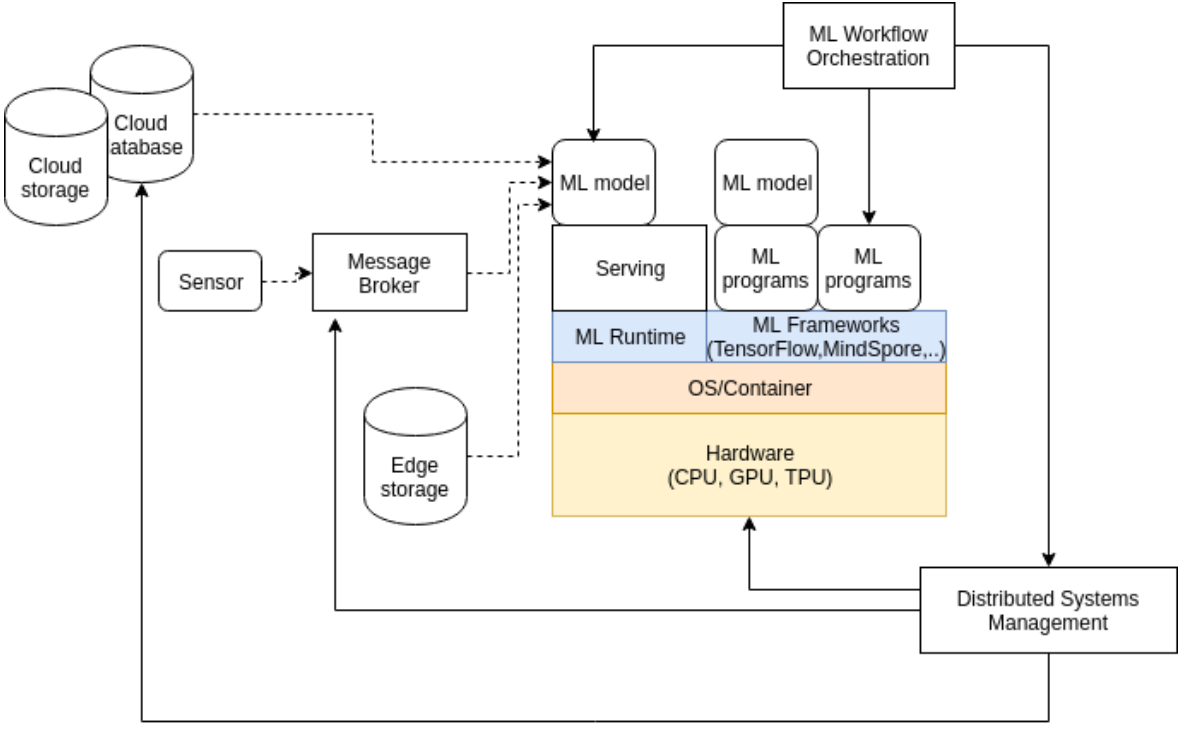


Interaction models



Software systems for ML in the edge

- **What are key features for ML runtime and programming frameworks?**
- **What are key features for resource management for running ML?**



Suitable ML and runtime for the edge: key requirements

- **Energy consumption**
- **Resource constraints**
  - less computation capabilities → precision and accuracy?
- **Latency and uncertainty**
- **Interfaces with different networks capabilities**
- **Support accelerators**
  - E.g., FPGA, AI Accelerators (e.g. Intel® Movidius Myriad X VPU)
- **Trade-offs between generic versus specific features**

Examples of ML frameworks and runtime for the edge

- **TF-lite (https://www.tensorflow.org/lite)**
- **https://github.com/Microsoft/EdgeML**
- **uTensor: https://github.com/uTensor/uTensor**
- **Androi NN (https://developer.android.com/ndk/guides/neuralnetworks)**
- **CoreML (https://developer.apple.com/machine-learning/core-ml/)**
- **PyTorch mobile (https://pytorch.org/mobile/home/)**
- **Snapdragon Neural Processing Engine SDK**
  - https://developer.qualcomm.com/docs/snpe/overview.html

Changes in MLOps

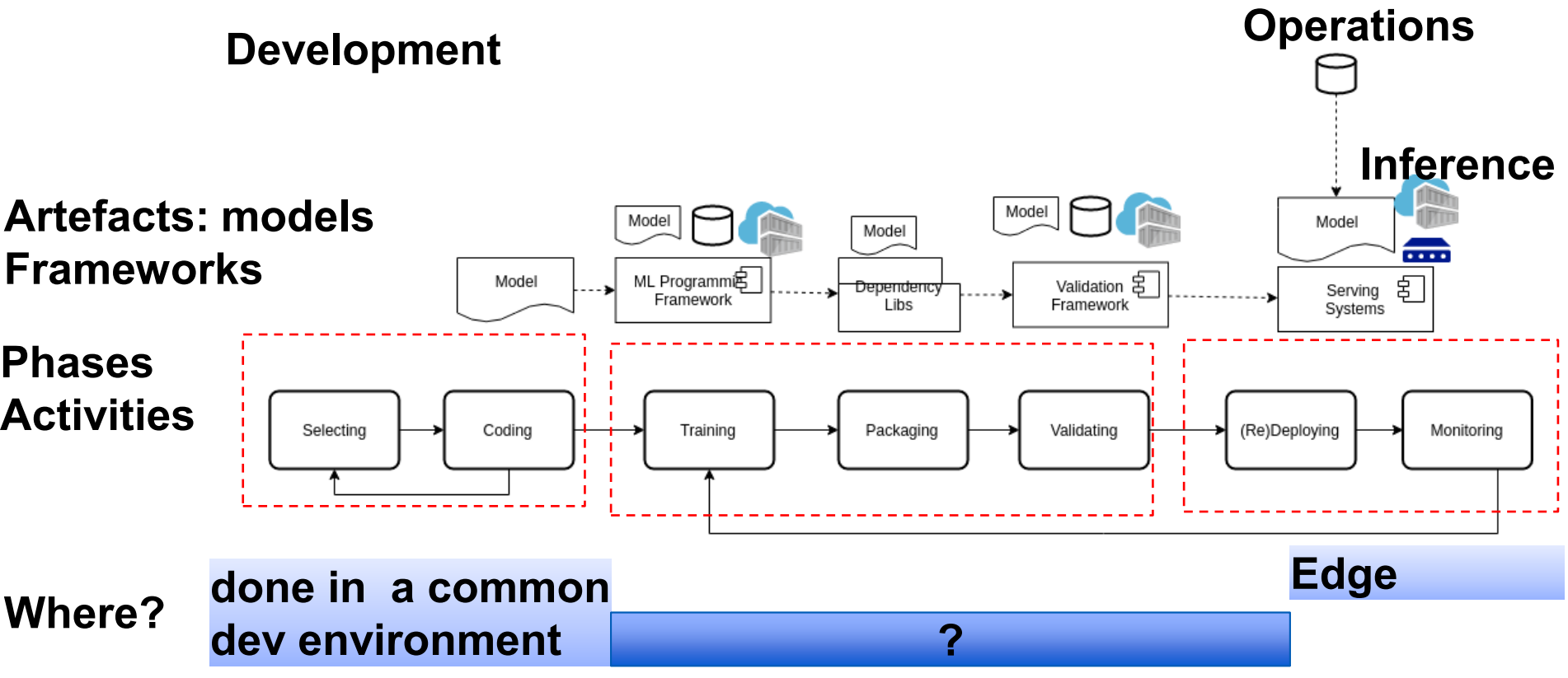
- **MLOps (ML DevOps)**
  - DevOps principles for ML
  - in ML engineering processes: key artefacts are ML models, data and runtime libs
- **Changes in ML with edge systems**
  - DevOps and DataOps activities in the edge
  - optimization and training activities
  - testing and benchmarks
  - monitoring

Example of MLOps

https://cloud.google.com/solutions/machine-learning/mlops-continuous-delivery-and-automation-pipelines-in-machine-learning

Is it the same in the edge?

MLOps in edge systems



Train in clouds/on-premise but edge deployment

- **Training in cloud and/or on-premise, and inferences in the edge**
  - issues of optimization, loss in transferring/conversion
  - accuracy loss due to the conversion
  - finding suitable edge machines for deployment (e.g., dynamic ML provisioning)
- **Training and inferences in the edge**
  - difficult with tools and resources
  - accuracy loss due to the (limited) training

Training in cloud and inference in the edge

https://blogs.gartner.com/paul-debeasi/files/2019/01/Train-versus-Inference.png

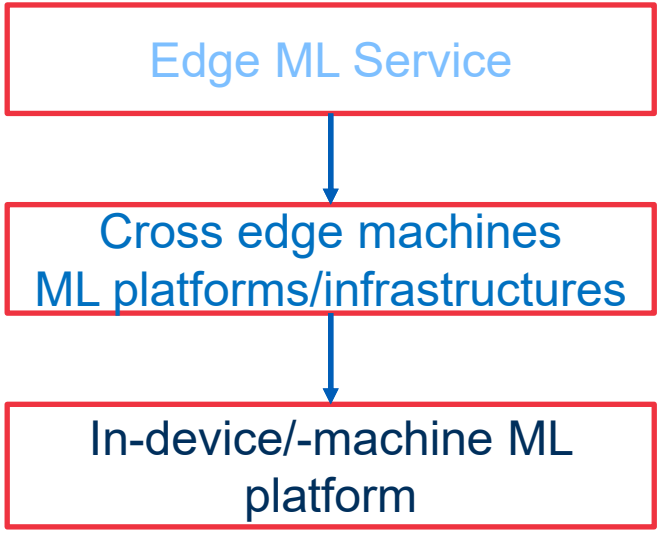
https://developer.qualcomm.com/docs/snpe/overview.html

Aalto University
School of Science

Some current research/engineering optimization problems

Multiple levels of optimization

Scope/level of abstraction



Research issues

ML serving, ML elasticity, ML observability and policies

ML function partitioning, distributed computation, orchestration, provisioning/(deployment, monitoring ..

Device-machine specific optimization

Aalto University
School of Science

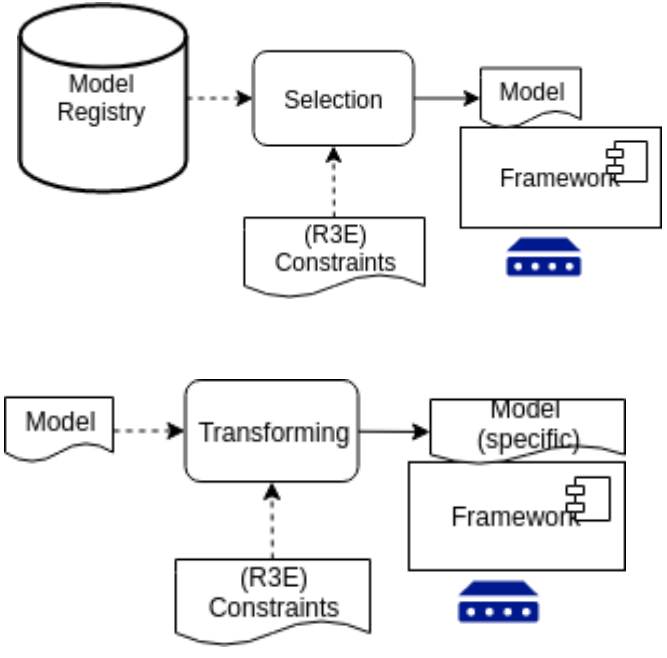
Our focus: to research and practice ML engineering analytics

Selected problems: transfer learning

- **Transfer learning**
  - repurpose a model trained for a task for another task
  - optimize an existing model for a new task
  - need model selection, reuse and model retraining
- **Transfer learning for the edge**
  - conversion/translation: transforming typical models in common (cloud) environments to edge models
  - symbiotic engineering: learning with simulations and inference with real data
  - application domains adaptation: adapt models among application domains

Selected problems: model selection and conversion

- **Model management and selection**
  - precision and time tradeoffs with computational requirements
  - work with microcontrollers and accelerators
- **Transforming**
  - a model can be supported by different frameworks
- **How will these issues affect Robustness and Reliability?**



Example: model conversion

- **Conversion**
  - just a simple form of “transforming”
- **A model fits into a single device/machine or into a set of machines?**
- **Single device/machine → no distributed computing**
  - focus on ML service and in-device optimization levels
- **A set of machines:**
  - which are distributed computing models for ML across machines? Ensemble and composition? horizonal scaling?

Selected problems: model optimization

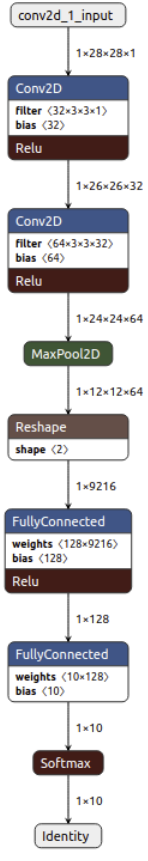
- **Pruning**
  - prune graphs for training, remove features in ML models which are not significant
- **Quantization**
  - reduce precision representation, storage, bandwidth
- **Conditional computation/Regularization**
  - activate certain units of the model
- **How will these issues affect Robustness, Reliability and Elasticity?**

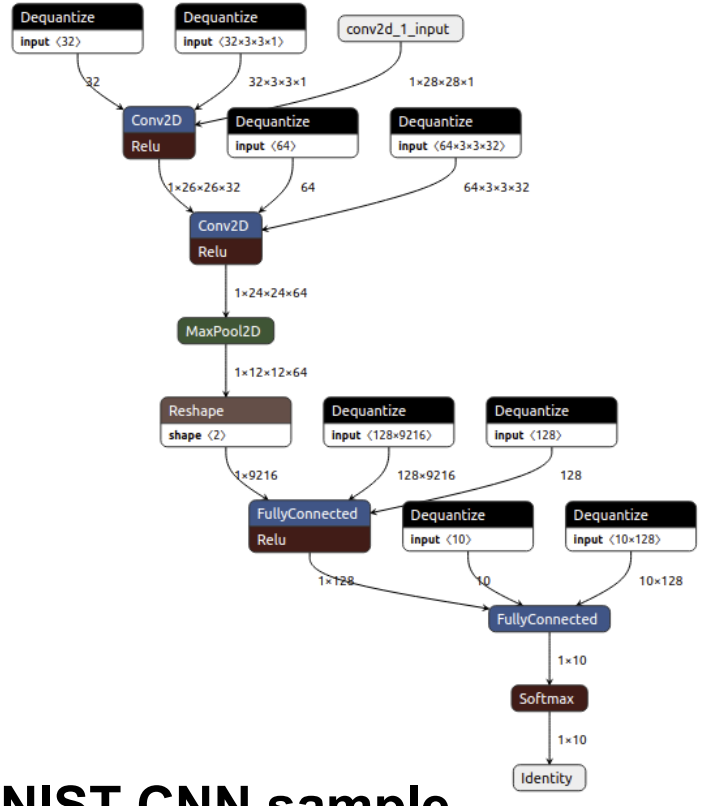
Tools/frameworks → "the ML compiler/optimize"

- **ONNX (Open Neural Network Exchange) format**
  - can be used as an intermediate representation compiled by tools to specific targets
- **Nvidia TensorRT**
  - JetPack SDK (https://developer.nvidia.com/embedded/jetpack)
- **OpenVINO (https://docs.openvinotoolkit.org/latest/index.html)**
- **Apache TVM (https://tvm.apache.org/)**
  - VTA (Versatile Tensor Accelerator)
- **Glow: https://github.com/pytorch/glow**
- **NNI: https://github.com/microsoft/nni/**

Example of Quantification by reducing floating point

32 bit floating point **16 bit floating point**





MNIST CNN sample

Conversion: the case of distributed models

- **Goal:**
  - if you have a model, now how to split it into edge/cloud?
- **Possible approaches**
  - partitioned model: split a model into different sub models
  - distributed ML networks: distribute the model graph across edge/cloud systems
  - federated learning: distributed training parts
  - chain of distributed ML models
- **Not a simple task – need to combine many techniques**

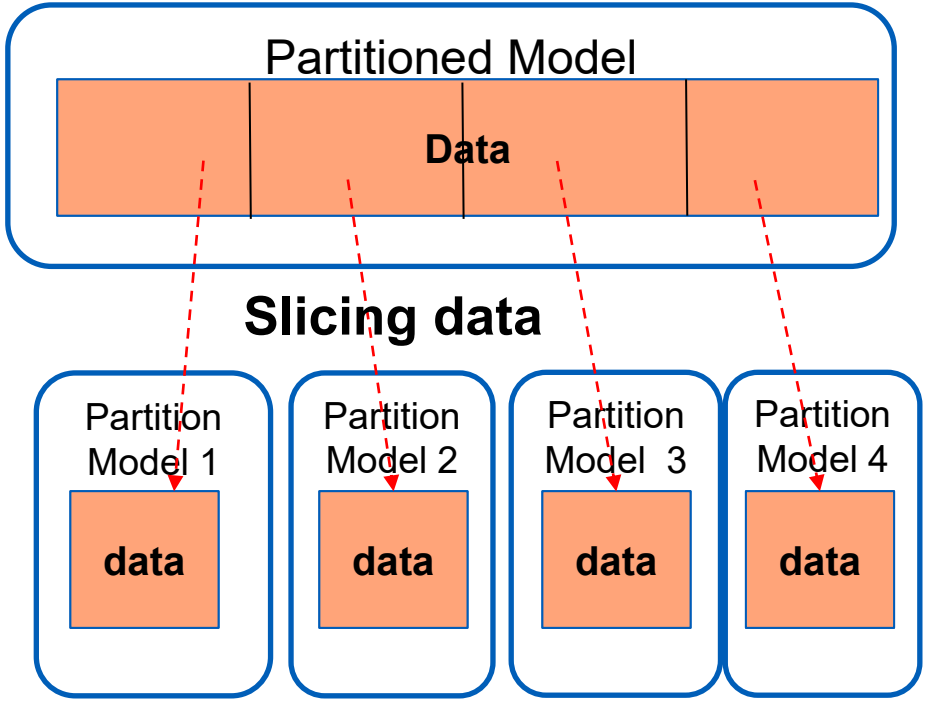
Partitioned models

- **A kind of "function partitioning" problems**
- **Training many partition/sub models, each for a partition data**
  - e.g., network operations in a city versus in country sides
  - a partitioned model consists of multiple sub models
    - *Work as a single model*
- **Slice input data into partitions, data in a suitable partition will be mapped into partition models (e.g., data partition)**

- **We can have a partitioned model running in multiple edges (e.g., each edge hosts a partition model)**

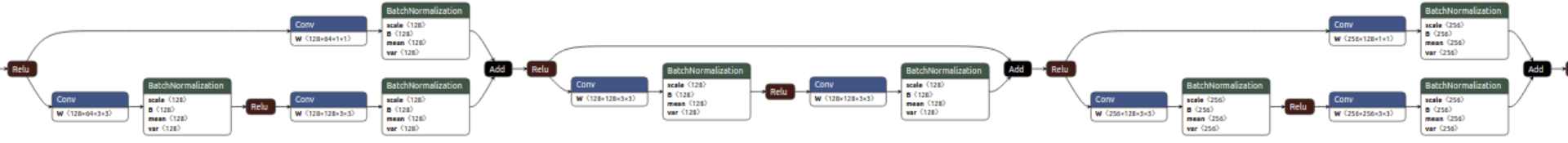
Partitioned models

- How to manage sub models for a partitioned model
- How to slice data for training and for inferences
- How to encapsulate complex runtime aspects to enable “virtualized” partitioned model serving



Distributed ML graph

Assume that you can partition a complex ML graph, what could be possible issues?

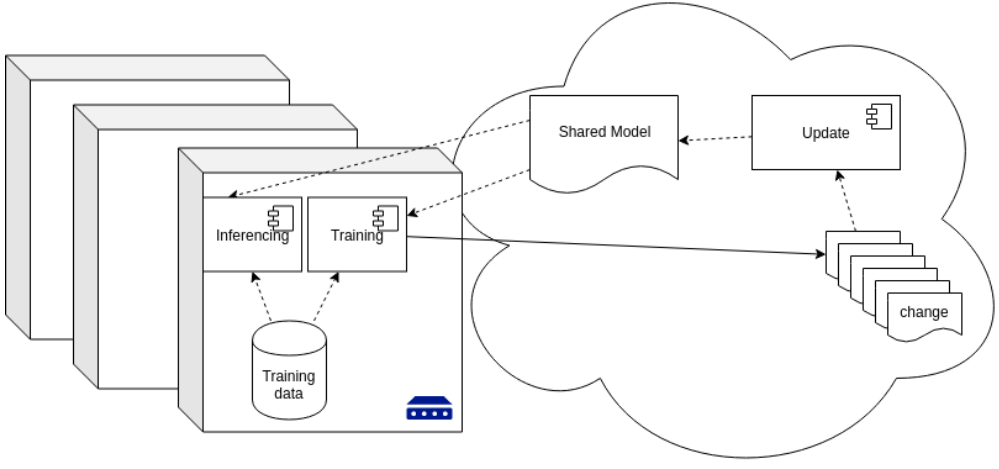


Edge device ⟷ Edge system ⟷ Cloud

How to partition? What would be the exchanges among subsystems

Selected problems: federated/distributed training with edges

Decentralized with a distributed set of devices holding data and carrying out (sub) training/inferencing



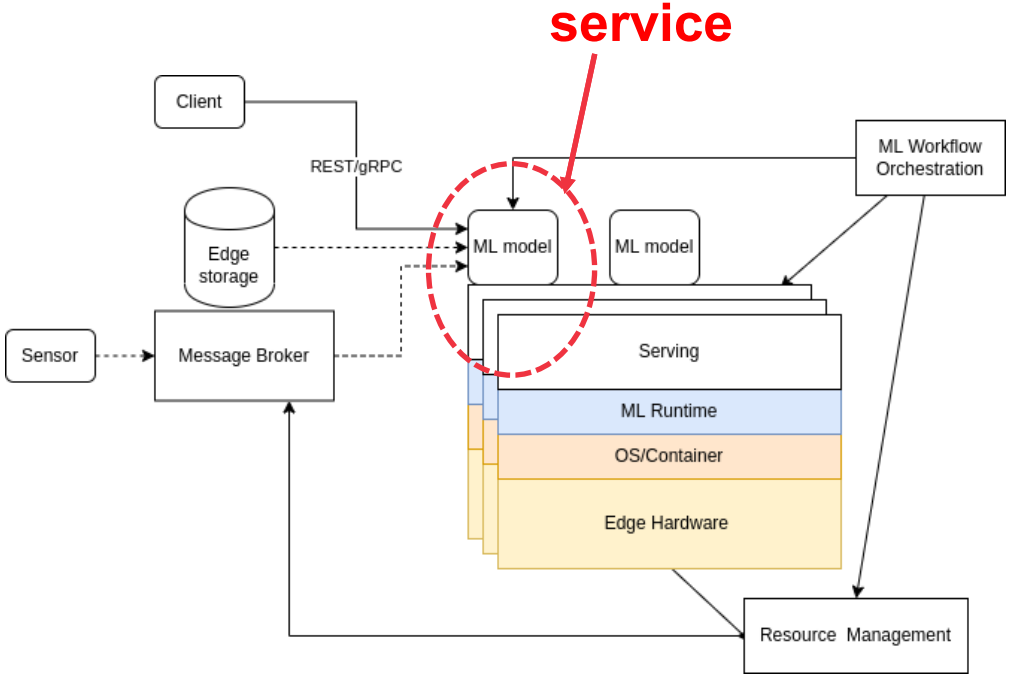
- **What about R3E?**
  - consensus in updates, secured aggregation protocols, reliability and elasticity

Some tools (not all for edge)

- **Some tools**
  - https://github.com/nttcslab/edge-consensus-learning
  - https://github.com/FederatedAI/FATE
  - https://github.com/tensorflow/federated
  - https://github.com/OpenMined/PySyft
  - https://github.com/horovod/horovod
  - https://flower.dev/
- **Key issues**
  - communications and task distributions
  - resource management

Selected problems: ML serving and pipeline design

- **ML Serving (and R3E)**
  - which dynamic service models we could have?
  - how to distribute tasks in model serving?
  - how to partition ML tasks in both edge and cloud?



(Near) real-time ML

- **Inferences at near real-time vs continual training**
  - how to make everything works in near real-time?
- **Inferences:**
  - inference serving is integrated into real time data pipeline and decision making
    - *ML serving design would be different due to performance and hosting environment, concurrency capacities, etc.*
- **Training (continual learning) → continuous ML (runtime adaptation)**
  - data flows to continual training in near real-time → features/labels are updated real-time for training → data transformation, feature extraction → (new, better) model management and serving

Study log

- **No study log but read papers and do the hands-on tutorial**
- **You can pickup some issues mentioned as the topic for your individual project**
  - Or incorporate some ideas into your individual project

- **ML with edge systems will increasingly be developed for many advanced software systems!**
  - Good areas for master theses/research projects.

Thanks!

Hong-Linh Truong
Department of Computer Science

rdsea.github.io